FINM 37601: Mathematical Market Microstructure

Assignment 4 – Anomaly Detection on Order Book Snapshots using Generative Adversarial Network (GAN)

Due Date: 11:59pm, Dec. 6, 2025

Please note: This assignment is "group-based"; each group can have up to 3 persons; please sign up for your group in the "People -> Group" section on canvas.

Introduction: With the advancements of AI technologies, market microstructure research is adopting more machine learning (ML) and deep learning (DL) techniques and tools to study trading data and market behaviors. One major area that has both academic merits and industry practicalities is "anomaly detection" of various market microstructure variables, esp. on order submissions by market participants. In fact, regulators in many countries have started using ML/DL-based monitoring tools to detect order flow anomalies, aiming at capturing adverse trading behaviors that may potentially disrupt market operations. In addition, "abnormal" order book behaviors – when detected correctly – can also be used as indicators for short-term price movement prediction.

In this assignment, we will use Generative Adversarial Network (GAN) technique to analyze a few selected stocks' order book snapshots to understand if "abnormal snapshots" exist in the market data. In addition, we want to determine that, when "abnormal snapshots" are detected, whether market microstructure variables (such as bid-ask spread, trade size, price returns, etc.) are significantly different from the time periods when no "abnormal snapshots" exist.

Generative Adversarial Network (GAN) Basics: a GAN is a model that belongs to the family of so-called "implicit generative models" in that it does not explicitly model the likelihood function nor provide a means for finding the latent variable corresponding to a given sample. A typical GAN is composed of two neural networks: a generator NN and a discriminator NN. A typical training process to create a GAN model has the following steps:

1. First, a set of randomized samples are input into a generator network, which

in turn produces “faked” outputs.

2. Second, the discriminator takes in both the “faked” outputs from step 1 above and a set of true samples, then produce two sets of results: one for the “faked” outputs from step 1, and the other for the true samples.
3. Third, two losses are calculated: one for the “faked” outputs that originally come from the generator, and the other for the true samples; these two losses are “backpropagated” to the generator and the discriminator, respectively, so that both the generator and the discriminator can adjust their parameters and hopefully produce smaller losses for the next run.
4. The above three steps are repeated numerous times until an equilibrium is reached. The reason why it is an “adversarial” network is because the generator originally wants to produce “faked” outputs; however, the discriminator – to put it simply – does not like “faked” outputs as long as it can detect them; therefore, to reach a final equilibrium, the generator has to be very good at producing “faked” outputs which are so close to the true samples (in statistical sense) that the discriminator cannot detect.

Once a GAN is produced, it can be applied in many situations. For example, the generator of trained GAN can be used to generate synthetic order book data based on a small sample of true data, and such synthetic order book data can be used to simulate specific trading strategies under various pre-defined market conditions. Also, the discriminator of a trained GAN can be used to filter out a few true samples that can be so different from most of the samples, and this is the “abnormal detection” application scenario.

Model Training and Testing: In this project, we will use minutely data of order books of a few selected stocks to conduct the analysis. Three sets of files are provided as the starting point for your implementation of a GAN model. These files are:

- `LOB_GAN_training.py`: This file defines the generator and discriminator classes and train the GAN model. The file takes in one stock’s three-month

order book data, partition it into the training set and the validation set, then train the model using Adam optimizer. Once the model is trained, it will be saved onto local drive.

- `LOB_GAN_testing.py`: This file loads in the trained GAN model, constructs testing data (the next three-month order book data), and (among other tasks) evaluates the testing data using the `eval()` function of the discriminator to find out if there is any "abnormal" order book snapshot.
- Six monthly order book files of three stocks (tickers are: 0050, 0056 and 2330) packaged into a zip file. These order book files include event level order book updates as well as transaction information (trade prices and sizes). The provided Python files do not use event level order book snapshots; instead, the files down-sample the data into minutely order book snapshots. Then, the first three months (202310, 202311, 202312) are used as training/validation data, and the next three months (202401, 202402, 202403) are used as testing data. This, of course, can be altered in your own analysis.

Question 1: (30 points) Test the two python files with the given orderbook snapshot data files. Make sure both Python files can successfully run on your local machine. As a result, you should be able to construct three generator models and discriminator models for the three stocks, respectively. Plot the change of loss function values (for both the training data set and the validation data set) for each training. Comment on your model training experience, such as which set of parameters that you modify to achieve desired training results. Hint: you may want to try different learning rate parameters for the generator and the discriminator for different stocks.

Note: You may need to install certain Python packages to run the python files. Check the top of the files to see which packages you may need to install/import. If you prefer to develop your own GAN for this assignment, either building on top of the provided Python files or developing anew, please do include your code in the final submission and provide as many comments in your code as you can so that our graders can do their best to understand your implementation.

Question 2: (50 points) Use the `LOB_GAN_testing.py` file to determine, in the testing data files of each stock, what order book snapshots can be good candidates for "abnormal" snapshots and constitute the "abnormal" set. The rest of the samples will constitute the "normal" set. Then, for each of the market microstructure variables below, compare the variable's statistical properties (such as mean, variance, skew, kurtosis, etc.) between the "abnormal" set and the "normal" set:

- Trade price returns and mid-quote returns.
- Trade size.
- Bid-ask spread and its first order difference.
- 1-level order book pressure as defined as $(BV_1 - SV_1)/(BV_1 + SV_1)$.
- 5-level order book pressure as defined as $\sum_{i=1}^{5}(BV_i - SV_i)/\sum_{i=1}^{5}(BV_i + SV_i)$.

Conduct Kolmogorov-Smirnov test on the distributions of each of the above market microstructure variables in the "abnormal" set and the "normal" set to determine if the respective variables have statistically different distributions between the two sets. Finally, based on the above statistical analysis between the "abnormal" set and the "normal" set of order book snapshots, comment on which one or a few of the above market microstructure variables may potentially contribute to the appearance of "abnormal" behaviors of order books, and how they can be potential high-frequency signals for short-term stock return forecasting.

Question 3: (20 points) Utilize the generator from the GAN model, generate a sequence of synthetic order book data for a few trading days in the testing data, compare the shape of the synthetic order books with those of the real order books. Comment on whether you trust the synthetic order books generated by the GAN model. Please be as specific and detailed as possible.

Final note: Each group submit your analysis results in one report, preferably in a "term paper" format with title, author list, abstract, and other necessary sections (such as introduction, analysis steps, results, conclusions, etc.). Final report should

not exceed 15 pages in total. Please state clearly your analysis process and results. Attach all supporting materials (code snippets, data files/tables, etc.) that you consider helpful and necessary to support your analysis and/or arguments.

End of assignment 4.